

# Package ‘SpatialTools’

July 18, 2023

**Type** Package

**Title** Tools for Spatial Data Analysis

**Version** 1.0.5

**Author** Joshua French <joshua.french@ucdenver.edu>

**Maintainer** Joshua French <joshua.french@ucdenver.edu>

**Depends** R (>= 3.0.2)

**Imports** spBayes (>= 0.3.0), Rcpp

**LinkingTo** Rcpp (>= 0.9.10), RcppArmadillo (>= 0.3.0)

**Description** Tools for spatial data analysis. Emphasis on kriging. Provides functions for prediction and simulation. Intended to be relatively straightforward, fast, and flexible.

**License** GPL (>= 2)

**Encoding** UTF-8

**LazyData** true

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2023-07-18 20:50:02 UTC

## R topics documented:

coincident . . . . .	2
cov.sp . . . . .	3
cov.st . . . . .	5
decomp.cov . . . . .	7
dist1 . . . . .	8
dist2 . . . . .	8
get.contours . . . . .	9
krige.ok . . . . .	10
krige.sk . . . . .	12
krige.uk . . . . .	14
maxlik.cov.sp . . . . .	16
maxlik.cov.st . . . . .	18

plot.contourLines . . . . .	20
rcondnorm . . . . .	21
rmvnorm . . . . .	23
simple.cov.sp . . . . .	24
simple.cov.time . . . . .	25
spLMPredictJoint . . . . .	26
toydata . . . . .	28

<b>Index</b>	<b>29</b>
--------------	-----------

---

coincident	<i>Determine coincident coordinates</i>
------------	---

---

### Description

coincident takes the coordinate matrices `coords1` and `coords2` and returns the indices of the coincident coordinates of the two matrices.

### Usage

```
coincident(coords1, coords2)
```

### Arguments

<code>coords1</code>	An $n1 \times 2$ numeric matrix of coordinates.
<code>coords2</code>	An $n2 \times 2$ numeric matrix of coordinates.

### Details

This function calls a compiled C++ program created using the Rcpp package. This (may) result in a significant speedup over pure R code since the search algorithm involves loops. We assume that there are no duplicate coordinates in `coords1` and `coords2`, respectively. In other words, each row of `coords1` is unique and each row of `coords2` is unique. There is at most 1 row of `coords1` that will match with a row in `coords2`.

### Value

Returns a matrix with the indices of the coincident locations. Specifically, an  $r \times 2$  matrix will be returned, with  $r$  being the number of coordinates in `coords1` coinciding with coordinates in `coords2`. If row  $i$  of the matrix is  $c(2, 5)$ , then the  $i$ th set of coincident locations is between the 2nd row of `coords1` and the 5th row of `coords2`. If there are no coincident locations, then a matrix of size  $0 \times 2$  is returned.

### Author(s)

Joshua French

**Examples**

```
#Generate two sets of coordinates
loc1 <- as.matrix(expand.grid(seq(0, 1, len = 25), seq(0, 1, len = 25)))
loc2 <- as.matrix(expand.grid(seq(0, 1, len = 101), seq(0, 1, len = 101)))
coincident(loc1, loc2)
```

cov.sp

*Calculates spatial covariance***Description**

Calculates spatial covariance matrix of the observed responses, and possibly, the responses to be predicted. If pcoords is not provided, then only V, the covariance matrix of the observed responses will be returned. If pcoords is provided, then Vp and Vop (the covariance matrix for predicted responses and between observed and predicted responses, respectively) will also be returned.

**Usage**

```
cov.sp(coords, sp.type = "exponential",
        sp.par = stop("specify sp.par argument"),
        error.var = 0, smoothness = 0.5, finescale.var = 0,
        pcoords = NULL, D = NULL, Dp = NULL, Dop = NULL)
```

**Arguments**

coords	A numeric matrix of size $n \times d$ containing the observed data locations.
sp.type	A character vector specifying the spatial covariance type. Valid types are currently exponential, gaussian, matern, matern2, and spherical.
sp.par	A vector of length 2 specifying the scale and strength of dependence of the covariance function. The first element is the variance of the underlying spatial process (also known as the hidden or latent spatial process). This value is also called the partial sill. The second element is the strength of dependence between responses.
error.var	A non-negative number indicating the variance of the error term.
smoothness	A positive number indicating the variance of the error term.
finescale.var	A non-negative positive number indicating the finescale variability. The is also called the microscale variance
pcoords	A numeric matrix of size $np \times d$ containing the locations of the responses to be predicted.
D	The Euclidean distance matrix for the coords matrix. Must be of size $n \times n$ .
Dp	The Euclidean distance matrix for the pcoords matrix. Must be of size $np \times np$ .
Dop	The Euclidean intersite distance matrix between the locations in coords and the locations in pcoords. Must be of size $n \times np$ .

## Details

The spatial covariance functions are parameterized in a manner consistent with the `cov.spatial` function in the `geoR` package. The `matern2` covariance function is an alternative covariance function suggested by Handcock and Wallis (1994). The benefit of this parameterization is that the range parameter is that it allows the effective range to be less dependent on the smoothness parameter.

The `D`, `Dp`, and `Dop` arguments are supplied to decrease the number of necessary computations needed when performing repetitive analysis or simulations. It is probably in the user's interest to not supply these arguments unless the duration of analysis is an important consideration. Note that these arguments override the information given in `coords` and `pcoords`, i.e., if `dist1(coords) != D`, then `D` is used in subsequent calculations, etc. This could create problems.

## Value

Returns a list with the following elements:

<code>V</code>	The covariance matrix for the observed responses. Will be of size $n \times n$ .
<code>Vp</code>	The covariance matrix for the predicted responses. Only returned if <code>pcoords</code> is supplied. Will be of size $np \times np$ .
<code>Vp</code>	The covariance matrix between the observed responses and the predicted responses. Only returned if <code>pcoords</code> is supplied. Will be of size $n \times np$ .

## Author(s)

Joshua French

## References

M.S. Handcock, J.R. Wallis. An approach to statistical spatial-temporal modeling of meteorological fields (with discussion). *Journal of the American Statistical Association*, 89 (1994), pp. 368–390.

## See Also

`simple.cov.sp`

## Examples

```
coords <- matrix(rnorm(30), ncol = 3)
cov.sp(coords = coords, sp.type = "exponential", sp.par = c(2, 1),
        error.var = 1)
```

---

cov.st	<i>Calculates spatio-temporal covariance</i>
--------	--

---

### Description

Calculates spatial covariance matrix of the observed responses, and possibly, the responses to be predicted. If pcoords is not provided, then only V, the covariance matrix of the observed responses will be returned. If pcoords is provided, then Vp and Vop (the covariance matrix for predicted responses and between observed and predicted responses, respectively) will also be returned.

### Usage

```
cov.st(coords, time, sp.type = "exponential",
        sp.par = stop("specify sp.par argument"),
        error.var = 0, smoothness = 0.5, finescale.var = 0,
        t.type = "ar1", t.par = .5,
        pcoords = NULL, ptime = NULL,
        D = NULL, Dp = NULL, Dop = NULL,
        T = NULL, Tp = NULL, Top = NULL)
```

### Arguments

coords	A numeric matrix of size $n \times d$ containing the observed data locations.
time	A numeric matrix of size $n \times 1$ containing the times at which the data was observed.
sp.type	A character vector specifying the spatial covariance type. Valid types are currently exponential, gaussian, matern, and spherical.
sp.par	A vector of length 2 specifying the scale and dependence of the covariance function. The first element refers to the variance of the hidden process (sometimes this is called the partial sill) while the second elements determines the strength of dependence between locations.
error.var	A non-negative number indicating the variance of the error term.
smoothness	A positive number indicating the variance of the error term.
finescale.var	A non-negative positive number indicating the finescale variability. The is also called the microscale variance
t.type	A character vector indicating the temporal dependance structure. Currently, only "ar1" is implemented.
t.par	A numeric vector of length 1 indicating the strength of temporal dependence.
pcoords	A numeric matrix of size $np \times d$ containing the locations of the responses to be predicted
ptime	A numeric matrix of size $np \times 1$ containing the times at which the responses are to be predicted.
D	The Euclidean distance matrix for the coords matrix. Must be of size $n \times n$ .

Dp	The Euclidean distance matrix for the pcoords matrix. Must be of size $np \times np$ .
Dop	The Euclidean intersite distance matrix between the locations in coords and the locations in pcoords. Must be of size $n \times np$ .
T	The Euclidean distance matrix for the time matrix. Must be of size $n \times n$ .
Tp	The Euclidean distance matrix for the ptime matrix. Must be of size $np \times np$ .
Top	The Euclidean intertime distance matrix between the times in time and ptime. Must be of size $n \times np$ .

### Details

At this point, this function only implements a separable spatio-temporal covariance function. If  $h$  is the distance between two sites, and  $t$  is the temporal lag between the times when the associated responses were observed, then the covariance function  $C(h, t) = C_s(h) \times C_t(t)$  where  $C_s$  is a spatial covariance function corresponding to the exponential, matern, gaussian, or spherical and  $C_t$  is the temporal covariance function corresponding to an ar1 process with  $C_t(t) = \phi^t$ .

The D, Dp, Dop, T, Tp, Top arguments are supplied to decrease the number of necessary computations needed when performing repetitive analysis or simulations. It is probably in the user's interest to not supply these arguments unless the duration of analysis is an important consideration. Note that these arguments override the information given in coords, pcoords, time, and ptime, i.e., if `dist1(coords) != D`, then D is used in subsequent calculations, etc. This could create problems.

### Value

Returns a list with the following elements:

V	The covariance matrix for the observed responses.
Vp	The covariance matrix for the predicted responses. Only returned if pcoords is supplied.
Vp	The covariance matrix between the observed responses and the predicted responses. Only returned if pcoords is supplied. Will be of size $n \times np$

### Author(s)

Joshua French

### See Also

`simple.cov.sp`

### Examples

```
coords <- matrix(rnorm(30), ncol = 3)
pcoords <- matrix(rnorm(90), ncol = 3)
time <- 1:10
ptime <- 1:30
cov.st(coords = coords, time = time, sp.type = "exponential",
       sp.par = c(2, 1), error.var = 1, t.type = "ar1", t.par = .5,
       pcoords = pcoords, ptime = ptime)
```

---

`decomp.cov`*Calculates decomposition of covariance matrix*

---

**Description**

Calculates a decomposition of the provided covariance matrix,  $V$ , using the chosen method.

**Usage**

```
decomp.cov(V, method = "eigen")
```

**Arguments**

<code>V</code>	A (symmetric, positive-definite) covariance matrix.
<code>method</code>	A character vector specifying the method used to decompose $V$ . Options are "eigen", "chol", or "svd" (Eigen decomposition, Cholesky decomposition, or Singular value decomposition, respectively).

**Details**

The matrix  $V$  is assumed to be symmetric and positive definite. Symmetry is checked, but the positive definiteness of the matrix is not. Returns a decomposition matrix  $U$  such that  $V = U \%*\% t(U)$ .

**Value**

Returns a decomposition matrix  $U$  such that  $V = U \%*\% t(U)$ .

**Author(s)**

Joshua French

**See Also**

`cov.sp`

**Examples**

```
data(toydata)
U <- decomp.cov(toydata$V, method = "chol")
#range(toydata$V - U %*% t(U))
```

---

dist1	<i>Calculate Euclidean distance matrix for a matrix of coordinates</i>
-------	--

---

**Description**

dist1 takes a matrix of coordinates and returns the Euclidean distance matrix of the coordinates. It does this using a compiled C program, so it is faster than the builtin R dist function.

**Usage**

```
dist1(coords)
```

**Arguments**

coords            An  $nr \times nc$  numeric matrix of coordinates.

**Value**

An  $nr \times nr$  matrix of Euclidean distances.

**Author(s)**

Joshua French

**See Also**

[dist](#), [dist2](#)

**Examples**

```
x <- matrix(rnorm(30), ncol = 3)
dist1(x)
```

---

dist2	<i>Calculate Euclidean distance matrix between coordinates of two matrices</i>
-------	--

---

**Description**

dist2 takes the matrices of coordinates coords1 and coords2 and returns the inter-Euclidean distances between coordinates.

**Usage**

```
dist2(coords1, coords2)
```



**Arguments**

coords1            An  $nr1 \times nc1$  numeric matrix of coordinates.  
coords2            An  $nr2 \times nc2$  numeric matrix of coordinates.

**Value**

An  $nr1 \times nr2$  matrix of Euclidean distances.

**Author(s)**

Joshua French

**See Also**

dist, dist1

**Examples**

```
x1 <- matrix(rnorm(30), ncol = 3)
x2 <- matrix(rnorm(60), ncol = 3)
dist2(x1, x2)
```

---

get.contours            *Extracts coordinates from contourLines function*

---

**Description**

Takes contours of contourLines function and extracts the associated coordinates.

**Usage**

```
get.contours(x)
```

**Arguments**

x                    A list returned by the contourLines function.

**Value**

Returns a 2-column matrix containing the coordinates making up the contours in contours.list.

**Author(s)**

Joshua French

**See Also**

contourLines, contour

**Examples**

```

data(volcano)
x <- 10*1:nrow(volcano)
y <- 10*1:ncol(volcano)
cL <- contourLines(x, y, volcano)
out <- get.contours(cL)

```

krige.ok

*Performs Ordinary Kriging***Description**

Performs Ordinary Kriging using  $y$ , the  $n \times 1$  matrix of observed responses,  $V$ , the (positive definite) covariance matrix of the observed responses,  $V_p$ , the  $np \times np$  covariance matrix of the responses to be predicted, and  $V_{op}$ , the  $n \times np$  matrix of covariances between the observed responses and the responses to be predicted.

**Usage**

```
krige.ok(y, V, Vp, Vop, nsim = 0, Ve.diag = NULL, method = "eigen")
```

**Arguments**

<code>y</code>	The vector of observed responses. Should be a matrix of size $n \times 1$ or a vector of length $n$ .
<code>V</code>	The covariance matrix of the observed responses. The size is $n \times n$ .
<code>Vp</code>	The covariance matrix of the responses to be predicted. The size is $np \times np$ .
<code>Vop</code>	The cross-covariance between the observed responses and the responses to be predicted. The size is $n \times np$ .
<code>nsim</code>	The number of simulated data sets to sample from the conditional predictive distribution.
<code>Ve.diag</code>	A vector of length $n$ specifying the measure error variances of the observed data. Only needed if <code>nsim &gt; 0</code> .
<code>method</code>	The method for decomposing $V$ in conditional simulation. Default is "eigen", for the Eigen decomposition. Alternatives are "chol" (Cholesky) and "svd" (Singular Value Decomposition).

**Details**

It is assumed that there are  $n$  observed data values and that we wish to make predictions at  $np$  locations.

If doing conditional simulation, the Cholesky decomposition should not work when there are coincident locations between the observed data locations and the predicted data locations. Both the Eigen and Singular Value Decompositions should work.

If user specifies `nsim` to be a positive integer, then `nsim` conditional realizations of the predictive distribution will be generated. If this is less than 1, then no conditional simulation is done. If `nsim` is a positive integer, then `Ve.diag` must also be supplied. `Ve.diag` should be a vector of length  $n$  specifying the measurement error variances of the observed data. This information is only used for conditional simulation, so this argument is only needed when `nsim > 0`. When conditional simulation is desired, then the argument `method` can be to specify the method used to decompose  $V$ . Options are "eigen", "chol", or "svd" (Eigen decomposition, Cholesky decomposition, or Singular value decomposition, respectively). This information is only used for conditional simulation, so this argument is only applicable when `nsim > 0`.

### Value

The function returns a list containing the following objects:

<code>pred</code>	A vector of length $np$ containing the predicted responses.
<code>mspe</code>	A vector of length $np$ containing the mean-square prediction error of the predicted responses.
<code>coeff</code>	A vector of length $k$ containing the estimated regression coefficients.
<code>vcov.coeff</code>	A $k \times k$ matrix containing the (estimated) covariance matrix of estimated the regression coefficients.
<code>simulations</code>	An $n \times nsim$ matrix containing the <code>nsim</code> realizations of the conditional realizations. Each column of the matrix represents a realization of the conditional normal distribution.

If `nsim > 0`, this list has class "krigeConditionalSample".

### Author(s)

Joshua French

### References

Statistical Methods for Spatial Data Analysis, Schabenberger and Gotway (2003). See p. 226-228.

### Examples

```
# create observed and predicted coordinates
ocoords <- matrix(runif(100), ncol = 2)
pcoords <- matrix(runif(200), ncol = 2)

# include some observed locations in the predicted coordinates
acoords <- rbind(ocoords, pcoords)

# create covariance matrix
C3 <- cov.sp(coords = ocoords, sp.type = "matern", sp.par = c(2, 1), smoothness = 1,
finescale = 0, error = 0.5, pcoords = acoords)

# generate data with error
y <- rmvnorm(nsim = 1, mu = rep(2, 50), V = C3$V) + rnorm(50, sd = sqrt(.5))
```

```
# use universal kriging to make predictions. Do not do conditional simulation
krige.obj <- krige.ok(as.vector(y), V = C3$V, Vp = C3$Vp, Vop = C3$Vop,
  nsim = 0)

#Do conditional simulation
krige.obj2 <- krige.ok(as.vector(y), V = C3$V, Vp = C3$Vp, Vop = C3$Vop,
  nsim = 100, Ve.diag = rep(.5, 50), method = "eigen")
```

---

krige.sk

*Performs simple kriging*


---

### Description

Performs simple kriging using  $y$ , a vector of length  $n$ ,  $V$ , the (positive definite) covariance matrix of the observed responses,  $V_p$ , the  $np \times np$  covariance matrix of the responses to be predicted,  $V_{op}$ , the  $n \times np$  matrix of covariances between the observed responses and the responses to be predicted, and  $m$ , a numeric vector of length 1 identifying the value of the mean for each response.

### Usage

```
krige.sk(y, V, Vp, Vop, m = 0, nsim = 0, Ve.diag = NULL, method = "eigen")
```

### Arguments

$y$	The vector of observed responses. Should be a matrix of size $n \times 1$ or a vector of length $n$ .
$V$	The covariance matrix of the observed responses. The size is $n \times n$ .
$V_p$	The covariance matrix of the responses to be predicted. The size is $np \times np$
$V_{op}$	The cross-covariance between the observed responses and the responses to be predicted. The size is $n \times np$ .
$m$	A numeric vector of length 1 giving the mean of each response.
$nsim$	The number of simulated data sets to sample from the conditional predictive distribution.
$Ve.diag$	A vector of length $n$ specifying the measure error variances of the observed data. Only needed if $nsim > 0$ .
$method$	The method for decomposing $V$ in conditional simulation. Default is "eigen", for the Eigen decomposition. Alternatives are "chol" (Cholesky) and "svd" (Singular Value Decomposition).

## Details

It is assumed that there are  $n$  observed data values and that we wish to make predictions at  $np$  locations. The mean is subtracted from each value of  $y$  before determining the kriging weights, and then the mean is added onto the predicted response.

If doing conditional simulation, the Cholesky decomposition should not work when there are coincident locations between the observed data locations and the predicted data locations. Both the Eigen and Singular Value Decompositions should work.

If user specifies `nsim` to be a positive integer, then `nsim` conditional realizations of the predictive distribution will be generated. If this is less than 1, then no conditional simulation is done. If `nsim` is a positive integer, then `Ve.diag` must also be supplied. `Ve.diag` is should be a vector of length  $n$  specifying the measurement error variances of the observed data. This information is only used for conditional simulation, so this argument is only needed when `nsim`  $>$  0. When conditional simulation is desired, then the argument `method` can be to specify the method used to decompose  $V$ . Options are "eigen", "chol", or "svd" (Eigen decomposition, Cholesky decomposition, or Singular value decomposition, respectively). This information is only used for conditional simulation, so this argument is only applicable when `nsim`  $>$  0.

## Value

The function returns a list containing the following objects:

<code>pred</code>	A vector of length $np$ containing the predicted responses.
<code>mspe</code>	A vector of length $np$ containing the mean-square prediction error of the predicted responses.
<code>simulations</code>	An $n \times nsim$ matrix containing the <code>nsim</code> realizations of the conditional realizations. Each column of the matrix represents a realization of the conditional normal distribution.
<code>mean</code>	The mean value ( <code>m</code> ) originally provided to the function

. If `nsim`  $>$  0, this list has class "krigeConditionalSample".

## Author(s)

Joshua French

## References

Statistical Methods for Spatial Data Analysis, Schabenberger and Gotway (2003). See p. 226-228.

## Examples

```
data(toydata)
y <- as.vector(toydata$y)
V <- toydata$V
Vp <- toydata$Vp
Vop <- toydata$Vop
krige.sk(y, V, Vp, Vop, m = 2)
```

krige.uk

*Performs universal kriging***Description**

Performs universal kriging using  $X$ , the  $n \times k$  design matrix for the regression coefficients of the observed data,  $y$ , the  $n \times 1$  matrix of observed responses,  $V$ , the (positive definite) covariance matrix of the observed responses,  $X_p$ , the  $np \times k$  design matrix of the responses to be predicted,  $V_p$ , the  $np \times np$  covariance matrix of the responses to be predicted, and  $V_{op}$ , the  $n \times np$  matrix of covariances between the observed responses and the responses to be predicted. If user specifies `nsim` to be a positive integer, then `nsim` conditional realizations of the predictive distribution will be generated.

**Usage**

```
krige.uk(y, V, Vp, Vop, X, Xp, nsim = 0, Ve.diag = NULL, method = "eigen")
```

**Arguments**

<code>y</code>	The vector of observed responses. Should be a matrix of size $n \times 1$ or a vector of length $n$ .
<code>V</code>	The covariance matrix of the observed responses. The size is $n \times n$ .
<code>Vp</code>	The covariance matrix of the responses to be predicted. The size is $np \times np$
<code>Vop</code>	The cross-covariance between the observed responses and the responses to be predicted. The size is $n \times np$
<code>X</code>	The design matrix of the observed data. The size is $n \times k$
<code>Xp</code>	The design matrix of the responses to be predicted. The size is $np \times k$
<code>.</code>	
<code>nsim</code>	The number of simulated data sets to sample from the conditional predictive distribution.
<code>Ve.diag</code>	A vector of length $n$ specifying the measure error variances of the observed data.
<code>method</code>	The method for decomposing $V$ in conditional simulation. Default is "eigen", for the Eigen decomposition. Alternatives are "chol" (Cholesky) and "svd" (Singular Value Decomposition).

**Details**

It is assumed that there are  $n$  observed data values and that we wish to make predictions at  $np$  locations. We assume that there are  $k$  regression coefficients (including the intercept). Both  $X$  and  $X_p$  should contain a column of 1's if an intercept is desired.

If doing conditional simulation, the Cholesky decomposition should not work when there are coincident locations between the observed data locations and the predicted data locations. Both the Eigen and Singular Value Decompositions should work.

If user specifies `nsim` to be a positive integer, then `nsim` conditional realizations of the predictive distribution will be generated. If this is less than 1, then no conditional simulation is done. If

`nsim` is a positive integer, then `Ve.diag` must also be supplied. `Ve.diag` should be a vector of length  $n$  specifying the measurement error variances of the observed data. This information is only used for conditional simulation, so this argument is only needed when `nsim > 0`. When conditional simulation is desired, then the argument `method` can be used to specify the method used to decompose  $V$ . Options are "eigen", "chol", or "svd" (Eigen decomposition, Cholesky decomposition, or Singular value decomposition, respectively). This information is only used for conditional simulation, so this argument is only applicable when `nsim > 0`.

## Value

The function returns a list containing the following objects:

<code>pred</code>	A vector of length $np$ containing the predicted responses.
<code>mspe</code>	A vector of length $np$ containing the mean-square prediction error of the predicted responses.
<code>coeff</code>	A vector of length $k$ containing the estimated regression coefficients.
<code>vcov.coeff</code>	A $k \times k$ matrix containing the (estimated) covariance matrix of estimated the regression coefficients.
<code>sim</code>	An $n \times nsim$ matrix containing the <code>nsim</code> realizations of the conditional realizations. Each column of the matrix represents a realization of the conditional normal distribution.

If `nsim > 0`, this list has class "krigeConditionalSample".

## Author(s)

Joshua French

## References

Statistical Methods for Spatial Data Analysis, Schabenberger and Gotway (2003). See p. 241-243.

## Examples

```
# create observed and predicted coordinates
ocoords <- matrix(runif(100), ncol = 2)
pcoords <- matrix(runif(200), ncol = 2)

# include some observed locations in the predicted coordinates
acoords <- rbind(ocoords, pcoords)

# create design matrices
X <- as.matrix(cbind(1, ocoords))
Xa <- as.matrix(cbind(1, acoords))

# create covariance matrix
C3 <- cov.sp(coords = ocoords, sp.type = "matern", sp.par = c(2, 1), smoothness = 1,
finescale = 0, error = 0.5, pcoords = acoords)

# set values of regression coefficients
```

```

coeff <- matrix(c(1, 2, 3), nrow = 1)

# generate data with error
y <- rmvnorm(nsim = 1, mu = tcrossprod(X, coeff), V = C3$V) + rnorm(50, sd = sqrt(.5))

# use universal kriging to make predictions. Do not do
# conditional simulation
krige.obj <- krige.uk(as.vector(y), V = C3$V, Vp = C3$Vp, Vop = C3$Vop,
X = X, Xp = Xa, nsim = 0)

#Do kriging with conditional simulation
krige.obj2 <- krige.uk(as.vector(y), V = C3$V, Vp = C3$Vp, Vop = C3$Vop,
X = X, Xp = Xa, nsim = 100,
Ve.diag = rep(.5, 50), method = "eigen")

```

---

maxlik.cov.sp

*Determines maximum likelihood estimates of covariance parameters*


---

## Description

Estimates covariance parameters of spatial covariance functions using maximum likelihood or restricted maximum likelihood. See `cov.sp` for more details of covariance functions to be estimated.

## Usage

```

maxlik.cov.sp(X, y, coords, sp.type = "exponential",
range.par = stop("specify range.par argument"),
error.ratio = stop("specify error.ratio argument"),
smoothness = 0.5,
D = NULL, reml = TRUE, lower = NULL, upper = NULL,
control = list(trace = TRUE), optimizer="nlminb")

```

## Arguments

<code>X</code>	A numeric matrix of size $n \times k$ containing the design matrix of the data locations.
<code>y</code>	A vector of length $n$ containing the observed responses.
<code>coords</code>	A numeric matrix of size $n \times d$ containing the locations of the observed responses.
<code>sp.type</code>	A character vector specifying the spatial covariance type. Valid types are currently exponential, gaussian, matern, and spherical.
<code>range.par</code>	An initial guess for the spatial dependence parameter.
<code>error.ratio</code>	A value non-negative value indicating the ratio <code>error.var/sp.par[1]</code> .
<code>smoothness</code>	A positive number indicating the smoothness of the matern covariance function, if applicable.
<code>D</code>	The Euclidean distance matrix for the <code>coords</code> matrix. Must be of size $n \times n$ .



reml	A boolean value indicating whether restricted maximum likelihood estimation should be used. Defaults to TRUE.
lower	A vector giving lower bounds for the covariance parameters <code>sp.par[2]</code> , <code>error.ratio</code> , and <code>smoothness</code> (when the model is matern). Order matters! If not given defaults to an upper bound of <code>Inf</code> for <code>sp.par[2]</code> , 1 for <code>error.ratio</code> , and 10 for <code>smoothness</code> .
upper	A vector giving upper bounds for the covariance parameters <code>sp.par[2]</code> , <code>error.ratio</code> , and <code>smoothness</code> (when the model is matern). Order matters! If not given defaults to an upper bound of <code>Inf</code> for <code>sp.par[2]</code> , 1 for <code>error.ratio</code> , and 10 for <code>smoothness</code> .
control	A list giving tuning parameters for the <code>nlmminb</code> function. See <code>nlmminb</code> for more details.
optimizer	A vector describing the optimization function to use for the optimization. Currently, only <code>nlmminb</code> is an acceptable value.

### Details

When doing the numerical optimization, the covariance function is reparameterized slightly to speedup computation. Specifically, the variance parameter for the process of interest, `sp.par[1]`, is profiled out, and the `error.var` parameter is parameterized as `sp.par[1] * error.ratio`, where `error.ratio = error.var/sp.par[1]`.

### Value

Returns a list with the following elements:

<code>sp.type</code>	The covariance form used.
<code>sp.par</code>	A vector containing the estimated variance of the hidden process and the spatial dependence.
<code>error.var</code>	The estimated error variance.
<code>smoothness</code>	The smoothness of the matern covariance function.
<code>par</code>	The final values of the optimization parameters. Note that these will not necessarily match <code>sp.par</code> , <code>error.var</code> , and <code>smoothness</code> because of the reparameterization.
<code>convergence</code>	Convergence message from <code>nlmminb</code> .
<code>message</code>	Message from <code>nlmminb</code> .
<code>iterations</code>	Number of iterations for optimization to converge.
<code>evaluations</code>	Evaluations from <code>nlmminb</code> .

### Author(s)

Joshua French

### See Also

`cov.st`

**Examples**

```
#generate 20 random (x, y) coordinates
coords <- matrix(rnorm(20), ncol = 2)

#create design matrix
X <- cbind(1, coords)

#create mean for observed data to be generated
mu <- X %>% c(1, 2, 3)

#generate covariance matrix
V <- exp(-dist1(coords))

#generate observe data
y <- rmvnorm(mu = mu, V = V)

#find maximum likelihood estimates of covariance parameters
maxlik.cov.sp(X = X, y = y, coords = coords,
  sp.type = "exponential", range.par = 1, error.ratio = 0,
  reml = TRUE)
```

---

maxlik.cov.st

*Determines maximum likelihood estimates of covariance parameters*


---

**Description**

Estimates covariance parameters of spatio-temporal covariance functions using maximum likelihood or restricted maximum likelihood. See `cov.st` for more details of covariance functions to be estimated. The covariance function is reparameterized slightly to speedup computation. Specifically, the variance parameter for the hidden process, `sp.par[1]`, is profiled out and the `error.var` parameter is parameterized as `sp.par[1] * error.ratio`.

**Usage**

```
maxlik.cov.st(X, y, coords, time, sp.type = "exponential",
  range.par = stop("specify range.par argument"),
  error.ratio = stop("specify error.ratio argument"),
  smoothness = 0.5, t.type = "ar1", t.par = .5, D = NULL, T = NULL,
  reml = TRUE, lower = NULL, upper = NULL, control = list(trace = TRUE),
  optimizer="nlminb")
```

**Arguments**

X	A numeric matrix of size $n \times k$ containing the design matrix of the data locations.
y	A vector of length $n$ containing the observed responses.
coords	A numeric matrix of size $n \times d$ containing the locations of the observed responses.

time	A numeric vector of length $n$ containing the time at which the responses were observed.
sp.type	A character vector specifying the spatial covariance type. Valid types are currently exponential, gaussian, matern, and spherical.
range.par	An initial guess for the spatial dependence parameter.
error.ratio	A value non-negative value indicating the ratio $\text{error.var}/\text{sp.par}[1]$ .
smoothness	A positive number indicating the variance of the error term.
t.type	A character vector indicating the spatial covariance type. Only ar1 is currently available.
t.par	A value specifying the temporal dependence parameter of the ar1 process.
D	The Euclidean distance matrix for the coords matrix. Must be of size $n \times n$ .
T	The Euclidean distance matrix for the time matrix. Must be of size $n \times n$ .
reml	A boolean value indicating whether restricted maximum likelihood estimation should be used. Defaults to TRUE.
lower	A vector giving lower bounds for the covariance parameters $\text{sp.par}[2]$ , $\text{error.ratio}$ , and $\text{smoothness}$ (when the model is matern). Order matters! If not given defaults to a lower bound of .001 for $\text{sp.par}[2]$ , 0 for $\text{error.ratio}$ , and .001 for $\text{smoothness}$ .
upper	A vector giving upper bounds for the covariance parameters $\text{sp.par}[2]$ , $\text{error.ratio}$ , and $\text{smoothness}$ (when the model is matern). Order matters! If not given defaults to an upper bound of Inf for $\text{sp.par}[2]$ , 1 for $\text{error.ratio}$ , and 10 for $\text{smoothness}$ .
control	A list giving tuning parameters for the nlmminb function. See nlmminb for more details.
optimizer	A vector describing the optimization function to use for the optimization. Currently, only nlmminb is an acceptable value.

### Details

When doing the numerical optimization, the covariance function is reparameterized slightly to speedup computation. Specifically, the variance parameter for the process of interest,  $\text{sp.par}[1]$ , is profiled out, and the  $\text{error.var}$  parameter is parameterized as  $\text{sp.par}[1] * \text{error.ratio}$ , where  $\text{error.ratio} = \text{error.var}/\text{sp.par}[1]$ .

### Value

Returns a list with the following elements:

sp.type	The covariance form used.
sp.par	A vector containing the estimated variance of the hidden process and the spatial dependence.
error.var	The estimated error variance.
smoothness	The smoothness of the matern covariance function.

par	The final values of the optimization parameters. Note that these will not necessarily match sp.par, error.var, and smoothness because of the reparameterization.
convergence	Convergence message from nlmminb.
message	Message from nlmminb.
iterations	Number of iterations for optimization to converge.
evaluations	Evaluations from nlmminb.

**Author(s)**

Joshua French

**See Also**

cov.st

**Examples**

```
#Generate locations and observed times
coords <- matrix(rnorm(40), ncol = 2)
time <- rep(1:2, each = 10)

#Calculate distance matrix for time vector
T <- dist1(matrix(time))

#create design matrix
X <- cbind(1, coords)

#create mean for observed data to be generated
mu <- X %*% c(1, 2, 3)

#generate covariance matrix for spatio-temporal data
V <- exp(-dist1(coords)) * .25^T

#generate observe data
y <- rmvnorm(mu = mu, V = V)

maxlik.cov.st(X = X, y = y, coords = coords, time = time,
  sp.type = "exponential", range.par = 1, error.ratio = 0,
  t.type = "ar1", t.par = .5, reml = TRUE)
```

---

plot.contourLines      *Plot contour lines*

---

**Description**

Plot contour lines from list produced by contourLines function.

**Usage**

```
## S3 method for class 'contourLines'
plot(x, begin=1, end = length(x), add = FALSE, ...)
```

**Arguments**

x	The list of contour lines (created by contourLines) you want to plot.
begin	Beginning position in list of contour lines you want to plot.
end	Ending position in list of contour lines you want to plot.
add	A boolean value indicating whether the contour lines should be added to an existing plot (add = TRUE) or should be plotted on a new plot (add = FALSE).
...	Additional arguments that will be passed to the plot or lines function.

**Value**

This function does not return anything; it only creates a new plot or modifies an existing plot.

**Author(s)**

Joshua French

**Examples**

```
data(volcano)
x <- 10*1:nrow(volcano)
y <- 10*1:ncol(volcano)
cL <- contourLines(x, y, volcano)
plot.contourLines(cL)
```

---

rcondnorm

*Generate from conditional normal distribution*


---

**Description**

Generates realizations from a multivariate normal distribution conditional on observed data vector

**Usage**

```
rcondnorm(nsim = 1, y, mu, mup, V, Vp, Vop, method = "eigen")
```

**Arguments**

<code>nsim</code>	An integer indicating the number of realizations from the distribution.
<code>y</code>	A vector of length <code>n</code> contained the observed data.
<code>mu</code>	The mean vector of the observed data. Should be a vector of length <code>n</code> .
<code>mup</code>	The mean vector of the responses to be generated. Should be a vector of length <code>np</code> .
<code>V</code>	The covariance matrix of the observed data. The matrix should be symmetric and positive definite. The size must be <i>ntimesn</i> .
<code>Vp</code>	The covariance matrix of the responses to be generated. The matrix should be symmetric and positive definite. The size must be <i>nptimesnp</i> .
<code>Vop</code>	The cross-covariance matrix between the observed data and the responses to be generated. The size must be <i>ntimesnp</i> .
<code>method</code>	The method for performing a decomposition of the covariance matrix. Possible values are "eigen", "chol", and "svd", Eigen value decomposition, Cholesky decomposition, or Singular Value Decomposition, respectively.

**Value**

An  $np \times nsim$  matrix containing the `nsim` realizations of the conditional normal distribution. Each column of the matrix represents a realization of the multivariate normal distribution.

**Author(s)**

Joshua French

**See Also**

`rmvnorm`

**Examples**

```
n <- 100
np <- 100

mu <- rep(1, 100)
mup <- rep(2, 100)

coords <- matrix(runif(2 * n), ncol = 2)
pcoords <- matrix(runif(2 * np), ncol = 2)

myV <- cov.sp(coords, sp.type = "exponential", c(1, 2), error.var = 1, pcoords = pcoords)

y <- rmvnorm(1, mu = mu, V = myV$V)

rcondnorm(3, y = y, mu = mu, mup = mup, V = myV$V, Vp = myV$Vp, Vop = myV$Vop, method = "chol")
```

---

`rmvnorm`*Generates realizations from a multivariate normal distribution*

---

**Description**

Generates realizations from a multivariate normal distribution.

**Usage**

```
rmvnorm(nsim = 1, mu, V, method = "eigen")
```

**Arguments**

<code>nsim</code>	An integer indicating the number of realizations from the distribution.
<code>mu</code>	A vector of length <code>n</code> containing the mean values of the multivariate normal distribution.
<code>V</code>	The covariance matrix of the multivariate normal distribution. The matrix should be symmetric and positive definite. The size must be <i>n</i> times <i>n</i> .
<code>method</code>	The method for performing a decomposition of the covariance matrix. Possible values are "eigen", "chol", and "svd", Eigen value decomposition, Cholesky decomposition, or Singular Value Decomposition, respectively.

**Value**

An  $n \times nsim$  matrix containing the `nsim` realizations of the multivariate normal distribution. Each column of the matrix represents a realization of the multivariate normal distribution.

**Author(s)**

Joshua French

**See Also**

`rmvnorm`

**Examples**

```
n <- 20
mu <- 1:n
V <- exp(-dist1(matrix(rnorm(n))))
rmvnorm(nsim = 100, mu = mu, V = V, method = "eigen")
```

---

`simple.cov.sp`*Calculates spatial covariance based on distance matrix*

---

### Description

Calculates a spatial covariance using a (Euclidean) distance matrix  $D$ . Not intended to be used directly by user (though it may be helpful to some). It is strongly recommended that you use the `cov.sp` function. No argument or error checking is provided for this function.

### Usage

```
simple.cov.sp(D, sp.type, sp.par, error.var, smoothness, finescale.var)
```

### Arguments

<code>D</code>	A distance matrix between locations
<code>sp.type</code>	A character vector specifying the spatial covariance type. Valid types are currently exponential, gaussian, matern, and spherical.
<code>sp.par</code>	A vector of length 2 specifying the scale and dependence of the covariance function. The first element refers to the variance of the hidden process (sometimes this is called the partial sill) while the second elements determines the strength of dependence between locations.
<code>error.var</code>	A non-negative number indicating the variance of the error term.
<code>smoothness</code>	A positive number indicating the variance of the error term.
<code>finescale.var</code>	A non-negative positive number indicating the finescale variability. The is also called the microscale variance

### Value

Returns a covariance matrix.

### Author(s)

Joshua French

### See Also

`~ cov.sp`

### Examples

```
coords <- matrix(rnorm(30), ncol = 3)
D <- dist1(coords)
simple.cov.sp(D = D, sp.type = "exponential", sp.par = c(2, 1),
error.var = 1, smoothness = 0.5, finescale.var = 0)
```



---

simple.cov.time	<i>Calculates temporal covariance based on distance matrix</i>
-----------------	--

---

**Description**

Calculates a temporal covariance using a (Euclidean) distance matrix *T*. Not intended to be used directly by user (though it may be helpful to some). It is used in the `covets` function. No argument or error checking is provided for this function.

**Usage**

```
simple.cov.time(T, t.type, t.par)
```

**Arguments**

<i>T</i>	A distance matrix.
<i>t.type</i>	A character vector specifying the temporal covariance type. Only "ar1" is currently implemented.
<i>t.par</i>	A vector of length 1 specifying the strength of dependence of the covariance function.

**Value**

Returns a covariance matrix.

**Author(s)**

Joshua French

**See Also**

`~ cov.st`

**Examples**

```
T <- dist1(matrix(1:10))
simple.cov.time(T = T, t.type = "ar1", t.par = .5)
```

---

spLMPredictJoint      *Returns posterior predictive sample from spLM object*

---

### Description

The function `spLMPredictJoint` collects posterior predictive samples for a set of new locations given a `spLM` object from the `spBayes` package.

### Usage

```
spLMPredictJoint(sp.obj, pred.coords, pred.covars, start = 1,
  end = nrow(sp.obj$p.theta.samples), thin = 1, verbose = TRUE, n.report = 100,
  noisy = FALSE, method = "eigen")
```

### Arguments

<code>sp.obj</code>	An <code>spLM</code> object returned by the <code>spLM</code> function in the <code>spBayes</code> package.
<code>pred.coords</code>	An $np \times 2$ matrix of $np$ prediction location coordinates in $R^2$ (e.g., easting and northing). The first column is assumed to be easting coordinates and the second column northing coordinates.
<code>pred.covars</code>	An $n \times p$ matrix of covariates matrix associated with the new locations.
<code>start</code>	Specifies the first sample included in the composition sampling.
<code>end</code>	Specifies the last sample included in the composition. The default is to use all posterior samples in <code>sp.obj</code> .
<code>thin</code>	A sample thinning factor. The default of 1 considers all samples between <code>start</code> and <code>end</code> . For example, if <code>thin = 10</code> then 1 in 10 samples are considered between <code>start</code> and <code>end</code> .
<code>verbose</code>	If TRUE, model specification and progress of the sampler is printed to the screen. Otherwise, nothing is printed to the screen.
<code>n.report</code>	The interval to report sampling progress.
<code>noisy</code>	If TRUE, then the posterior sample for the response is for the signal + error noise. The default, FALSE, assumes the user wants the error-free process.
<code>method</code>	Method used to decompose covariance matrix. Options are "chol", "eigen", and "svd" for the Cholesky, Eigen, and singular value decomposition approaches, respectively.

### Details

This function samples from the joint posterior predictive distribution of a Bayesian spatial linear model. Specifically, it is intended to be similar to the `spPredict` function in the `spBayes` except that it samples from the joint distribution instead of the marginal distribution. However, it will only work for `spLM` objects and should have the same limitations as the `spLM` and `spPredict` functions. Note that the `spRecover` function is called internally to recover the posterior samples from the posterior distribution of the spatial model.

**Value**

The function returns a  $np \times B$  matrix of posterior predictive samples, where B is the number of posterior samples. The class is `jointPredictiveSample`.

**Author(s)**

Joshua French

**See Also**

`spLM`, `spPredict`, `spRecover`

**Examples**

```
# Set parameters
n <- 100
np <- 12
n.samples <- 10
burnin.start <- .5 * n.samples + 1
sigmasq <- 1
tausq <- 0.0
phi <- 1
cov.model <- "exponential"
n.report <- 5

# Generate coordinates
coords <- matrix(runif(2 * n), ncol = 2);
pcoords <- as.matrix(expand.grid(seq(0, 1, len = 12), seq(0, 1, len = 12)))

# Construct design matrices
X <- as.matrix(cbind(1, coords))
Xp <- cbind(1, pcoords)

# Specify priors
starting <- list("phi" = phi, "sigma.sq" = sigmasq, "tau.sq" = tausq)
tuning <- list("phi" = 0.1, "sigma.sq" = 0.1, "tau.sq" = 0.1)
priors.1 <- list("beta.Norm" = list(c(1, 2, 1), diag(100, 3)),
               "phi.Unif" = c(0.00001, 10), "sigma.sq.IG" = c(1, 1))

# Generate data
B <- rnorm(3, c(1, 2, 1), sd = 10)
phi <- runif(1, 0, 10)
sigmasq <- 1/rgamma(1, 1, 1)
V <- simple.cov.sp(D = dist1(coords), cov.model, c(sigmasq, 1/phi), error.var = tausq,
smoothness = nu, finescale.var = 0)
y <- X %*% B + rmvnorm(1, rep(0, n), V) + rnorm(n, 0, sqrt(tausq))

# Create spLM object
library(spBayes)
m1 <- spBayes::spLM(y ~ X - 1, coords = coords, starting = starting,
tuning = tuning, priors = priors.1, cov.model = cov.model,
n.samples = n.samples, verbose = FALSE, n.report = n.report)
```

```
# Sample from joint posterior predictive distribution
y1 <- splMPredictJoint(m1, pred.coords = pcoords, pred.covars = Xp,
start = burnin.start, verbose = FALSE, method = "chol")
```

---

toydata

*A toy data set for use in examples.*

---

### Description

A list containing  $X$ , a  $50 \times 3$  design matrix,  $y$ , a vector of length 50 of observed responses,  $V$ , a  $50 \times 50$  covariance matrix for the observed data,  $X_p$ , a  $121 \times 3$  design matrix for the predicted responses,  $V_p$ , the  $121 \times 121$  covariance matrix of the predicted responses,  $V_{op}$ , the  $50 \times 121$  covariance matrix between the observed responses and the predicted responses,  $coords$ , a  $50 \times 2$  matrix containing the sites of the 50 observed responses, and  $pcoords$ , a  $121 \times 2$  matrix containing the 121 sites for the predicted responses (a  $11 \times 11$  regular grid over the domain  $[0, 1] \times [0, 1]$ ).

### Usage

```
data(toydata)
```

### Author(s)

Joshua French

### Examples

```
data(toydata)
```

# Index

- \* **bayesian**
  - spLMPredictJoint, 26
- \* **conditional**
  - rcondnorm, 21
- \* **contour lines**
  - get.contours, 9
- \* **contour**
  - get.contours, 9
- \* **covariance**
  - cov.sp, 3
  - cov.st, 5
  - decomp.cov, 7
  - maxlik.cov.sp, 16
  - maxlik.cov.st, 18
  - simple.cov.sp, 24
  - simple.cov.time, 25
- \* **decomposition**
  - decomp.cov, 7
- \* **joint**
  - spLMPredictJoint, 26
- \* **kriging**
  - krige.ok, 10
  - krige.sk, 12
  - krige.uk, 14
- \* **level curve**
  - get.contours, 9
- \* **maximum likelihood**
  - maxlik.cov.sp, 16
  - maxlik.cov.st, 18
- \* **multivariate**
  - rcondnorm, 21
  - rmvnorm, 23
- \* **normal**
  - rcondnorm, 21
  - rmvnorm, 23
- \* **ordinary kriging**
  - krige.ok, 10
- \* **ordinary**
  - krige.ok, 10
- \* **plot**
  - plot.contourLines, 20
- \* **prediction**
  - spLMPredictJoint, 26
- \* **simple kriging**
  - krige.sk, 12
- \* **simple**
  - krige.sk, 12
- \* **spatial**
  - cov.sp, 3
  - cov.st, 5
  - maxlik.cov.sp, 16
  - maxlik.cov.st, 18
  - simple.cov.sp, 24
  - simple.cov.time, 25
- \* **spatio-temporal**
  - cov.st, 5
- \* **time**
  - cov.st, 5
- \* **universal kriging**
  - krige.uk, 14
- \* **universal**
  - krige.uk, 14
- coincident, 2
- cov.sp, 3
- cov.st, 5
- decomp.cov, 7
- dist, 8
- dist1, 8
- dist2, 8, 8
- get.contours, 9
- krige.ok, 10
- krige.sk, 12
- krige.uk, 14
- maxlik.cov.sp, 16
- maxlik.cov.st, 18

`plot.contourLines`, 20

`rcondnorm`, 21

`rmvnorm`, 23

`simple.cov.sp`, 24

`simple.cov.time`, 25

`spLMPredictJoint`, 26

`toydata`, 28